Summer 2022 Data Science Intern Challenge

Question 1: Given some sample data, write a program to answer the following: click here to access the required data set

> On Shopify, we have exactly 100 sneaker shops, and each of these shops sells only one model of shoe. We want to do some analysis of the average order value (AOV). When we look at orders data over a 30 day window, we naively calculate an AOV of $3145.13. Given that we know these shops are selling sneakers, a relatively affordable item, something seems wrong with our analysis.
>
> **a.** Think about what could be going wrong with our calculation. Think about a better way to evaluate this data.

To help answer this question, I decided to use R as a way of exploring more of this dataset. The first step is to download any libraries I might need and then open the file itself.

```
library("knitr")
library("ggplot2")
library("kableExtra")
library("readr")
library("car")
library("readxl")
library(data.table)
library(tidyverse)

winterChallenge <- read_excel("winterChallengeDataset.xlsx")
```

From there, the first thing I think would be most useful is to see how the AOV of $3145.13 was calculated. My first intuition is to use the simplest formula for the average: the total order amount of all orders divided by the number of orders. Upon running the following code, we get the exact value.

```
AOV <- sum(winterChallenge$order_amount) / nrow(winterChallenge)
AOV
```

```{r}
AOV <- sum(winterChallenge$order_amount) / nrow(winterChallenge)
AOV
```

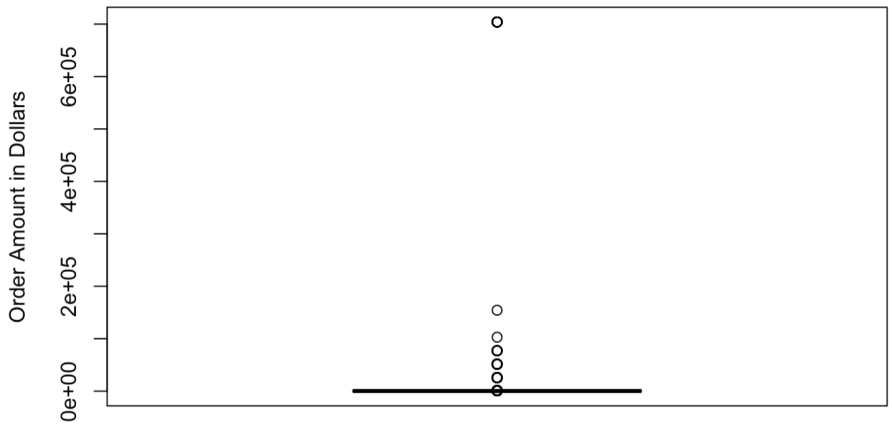
```
[1] 3145.128
```

The number of rows is pretty self explanatory; there's nothing that could be wrong with that. That means our problem is probably coming from the order_amount column. Let's do some exploration on that column.

```
summary(winterChallenge$order_amount)
boxplot(winterChallenge$order_amount, ylab = "Order Amount in Dollars")
```

When running these lines, we can get the summary of the data, and a boxplot of what the data looks like.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     90     163     284    3145     390  704000
```



As we can see in the summary statistics, our median is drastically lower than our mean, suggesting that there are outliers that cause that data to be skewed to the right. Our boxplot confirms this. My next thought is to explore two different things. The first is to look for a correlation between the high order values and the number of actual items bought in the order. The second is to see if there's any similarities in these super high volume orders.

```
cor(winterChallenge$order_amount, winterChallenge$total_items)
```

```
[1] 0.9917469
```

As we can see, the correlation between the order amount and the items in that order is almost 1, suggesting that the orders with a large order amount also have a very large amount of total items. Let's now look more at these high volume orders and see if we can see anything special about them.

```
newWinterChallenge <-
winterChallenge[order(-winterChallenge$order_amount),]
head(newWinterChallenge, n = 80)
```

Rows 1-10

order_id <dbl>	shop_id <dbl>	user_id <dbl>	order_amount <dbl>	total_items <dbl>	payment_method <chr>	created_at <S3: POSIXct>
16	42	607	704000	2000	credit_card	2017-03-07 04:00:00
61	42	607	704000	2000	credit_card	2017-03-04 04:00:00
521	42	607	704000	2000	credit_card	2017-03-02 04:00:00
1105	42	607	704000	2000	credit_card	2017-03-24 04:00:00
1363	42	607	704000	2000	credit_card	2017-03-15 04:00:00
1437	42	607	704000	2000	credit_card	2017-03-11 04:00:00
1563	42	607	704000	2000	credit_card	2017-03-19 04:00:00
1603	42	607	704000	2000	credit_card	2017-03-17 04:00:00
2154	42	607	704000	2000	credit_card	2017-03-12 04:00:00
2298	42	607	704000	2000	credit_card	2017-03-07 04:00:00

Rows 11-20

order_id <dbl>	shop_id <dbl>	user_id <dbl>	order_amount <dbl>	total_items <dbl>	payment_method <chr>	created_at <S3: POSIXct>
2836	42	607	704000	2000	credit_card	2017-03-28 04:00:00
2970	42	607	704000	2000	credit_card	2017-03-28 04:00:00
3333	42	607	704000	2000	credit_card	2017-03-24 04:00:00
4057	42	607	704000	2000	credit_card	2017-03-28 04:00:00
4647	42	607	704000	2000	credit_card	2017-03-02 04:00:00
4869	42	607	704000	2000	credit_card	2017-03-22 04:00:00
4883	42	607	704000	2000	credit_card	2017-03-25 04:00:00
692	78	878	154350	6	debit	2017-03-27 22:51:43
2493	78	834	102900	4	debit	2017-03-04 04:37:33
1260	78	775	77175	3	credit_card	2017-03-27 09:27:19

Rows 21-30

order_id <dbl>	shop_id <dbl>	user_id <dbl>	order_amount <dbl>	total_items <dbl>	payment_method <chr>	created_at <S3: POSIXct>
2565	78	915	77175	3	debit	2017-03-25 01:19:35
2691	78	962	77175	3	debit	2017-03-22 07:33:25
2907	78	817	77175	3	debit	2017-03-16 03:45:46
3404	78	928	77175	3	debit	2017-03-16 09:45:04
3725	78	766	77175	3	credit_card	2017-03-16 14:13:25
4193	78	787	77175	3	credit_card	2017-03-18 09:25:31
4421	78	969	77175	3	debit	2017-03-09 15:21:34
4716	78	818	77175	3	debit	2017-03-05 05:10:43
491	78	936	51450	2	debit	2017-03-26 17:08:18
494	78	983	51450	2	cash	2017-03-16 21:39:35

Rows 51-60

order_id <dbl>	shop_id <dbl>	user_id <dbl>	order_amount <dbl>	total_items <dbl>	payment_method <chr>	created_at <S3: POSIXct>
1453	78	812	25725	1	credit_card	2017-03-17 18:09:54
2271	78	855	25725	1	credit_card	2017-03-14 23:58:21
2549	78	861	25725	1	cash	2017-03-17 19:35:59
2774	78	890	25725	1	cash	2017-03-26 10:36:43
2923	78	740	25725	1	debit	2017-03-12 20:10:58
3086	78	910	25725	1	cash	2017-03-26 01:59:26
3152	78	745	25725	1	credit_card	2017-03-18 13:13:07
3441	78	982	25725	1	debit	2017-03-19 19:02:53
3781	78	889	25725	1	cash	2017-03-11 21:14:49
4041	78	852	25725	1	cash	2017-03-02 14:31:11

Rows 61-70

order_id <dbl>	shop_id <dbl>	user_id <dbl>	order_amount <dbl>	total_items <dbl>	payment_method <chr>	created_at <S3: POSIXct>
4506	78	866	25725	1	debit	2017-03-22 22:06:00
4585	78	997	25725	1	cash	2017-03-25 21:48:43
4919	78	823	25725	1	cash	2017-03-15 13:26:46
1365	42	797	1760	5	cash	2017-03-10 06:28:21
1368	42	926	1408	4	cash	2017-03-13 02:38:33
1472	42	907	1408	4	debit	2017-03-12 23:00:21
3539	43	830	1086	6	debit	2017-03-17 19:56:29
4142	54	733	1064	8	debit	2017-03-07 17:05:17
939	42	808	1056	3	credit_card	2017-03-13 23:43:45
2988	42	819	1056	3	cash	2017-03-03 09:09:25

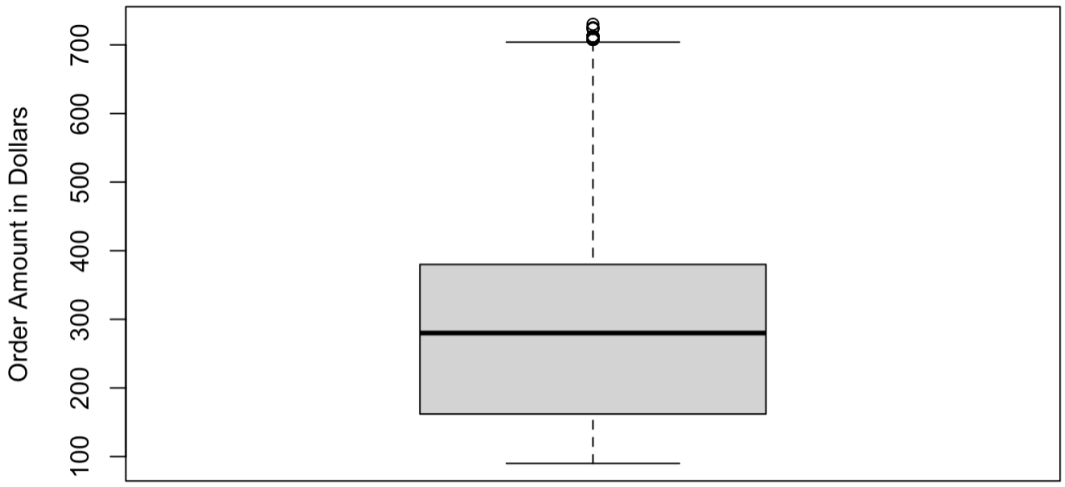
Two things immediately stand out. All the orders of $704000 are by the same user, and all the orders of $25725, $51450, and $77175 are by the same shop. Additionally, $25725, $51450, and $77175 are all multiples of one another. This suggests that shop 78 is selling an item at a price of $25725, and different people are buying different amounts of these shoes.

We've now identified our problem: we have outliers by a certain user and a certain store. Now, if we want to solve this problem, the best way would be to get rid of these outliers when looking at the order amount.

Although there is no set definition of what an outlier is, I learned in high school that an outlier is anything more than 1.5 times the interquartile range outside the 1st and 3rd quartile. As a result, that's what I decided to use here. I also pulled up a box plot and summary statistics on the new dataset without the outliers.

```
q1 <- quantile(winterChallenge$order_amount)[2]
q3 <- quantile(winterChallenge$order_amount)[4]
IQR <- q3 -q1

winterChallengeFiltered <- filter(winterChallenge, order_amount > q1 - (1.5
* IQR), order_amount < q3 + (1.5 * IQR))
boxplot(winterChallengeFiltered$order_amount, ylab = "Order Amount in
Dollars")
summary(winterChallengeFiltered$order_amount)
```




```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
90.0   162.0   280.0   293.7   380.0   730.0
```

As we can see, the data now has no extreme outliers, and is likely a more accurate representation of the AOV.

b. What metric would you report for this dataset?

While the mean is greatly improved because we got rid of the major outliers, as we can see above, the boxplot is still slightly skewed, with the maximum (730) being significantly farther from the median (280) than the minimum (90). As a result, the mean will still be more heavily affected by the higher values, so the metric I would report would be the **median.**

c. What is its value?

The median is **280.**

Question 2: For this question you'll need to use SQL. Follow this link to access the data set required for the challenge. Please use queries to answer the following questions. Paste your queries along with your final numerical answers below.

a. How many orders were shipped by Speedy Express in total?

After looking at the "Shippers" table, I was able to see that Speedy Express's ShipperID is 1. In order to get the result, I need to find the number of orders in the "Orders" table with ShipperID 1.

```
SELECT COUNT(ShipperID) AS 'Number of Orders by Speedy Express'
FROM [Orders]
WHERE ShipperID = 1
```

Number of Records: 1

Number of Orders by Speedy Express
54

The total orders shipped is 54.

A join function could also be used on the ShipperID in the "Orders: table and ShipperID in the "Shippers" table where the name is Speedy Express, but in order to do that, I would have to go look at the "Shippers" table regardless, and it'd be easier to just look and find what the ShipperID of Speedy Express is.

b. What is the last name of the employee with the most orders?

Looking at the "Orders" table, I see that every order has a specific employee ID with it. If I can get all the counts of orders by a specific employee ID, I can then see which employee ID has the highest amount of orders, and then find the last name. When we run the following query, we can get the table down below.

```
SELECT EmployeeID, COUNT(*) AS 'Number of Orders'
FROM [Orders]
GROUP BY 1
ORDER BY 2 DESC
```

Number of Records: 9

EmployeeID	Number of Orders
4	40
3	31
1	29
8	27
2	20
6	18
7	14
5	11
9	6

I can then look at the “Employee” table, and see that the last name of EmployeeID 4 is **Peacock**.

Similar to the first part, I could also do a join. It would look something like this:

```
SELECT [Employees].LastName, Count([Orders].EmployeeID) AS 'Number of
Orders'
FROM [Orders]
JOIN [Employees]
ON [Orders].EmployeeID = [Employees].EmployeeID
GROUP BY 1
ORDER BY 2 DESC
```

Number of Records: 9

LastName	Number of Orders
Peacock	40
Leverling	31
Davolio	29
Callahan	27
Fuller	20
Suyama	18
King	14
Buchanan	11
Dodsworth	6

However, similar to the first part, I find it to be easier to make the SQL queries simpler and just look for information that can be found almost instantly.

c. What product was ordered the most by customers in Germany?

This time, we need to join three separate tables: the “Customers” table with the “Orders” table so we can see the country of each customer that ordered something, and the “OrderDetails” table with the “Orders” table, so we can see the specific product and quantity of the product of each order.

```
SELECT [OrderDetails].ProductID, SUM([OrderDetails].Quantity) AS 'Total
Ordered'
FROM [Orders]
JOIN [Customers]
ON [Customers].CustomerID = [Orders].CustomerID
JOIN [OrderDetails]
ON [OrderDetails].OrderID = [Orders].OrderID
WHERE [Customers].Country = 'Germany'
GROUP BY 1
ORDER BY 2 DESC
```

Number of Records: 45

ProductID	Total Ordered
40	160
31	125
23	105
35	100
19	95
72	86
2	84
76	75
71	75
60	75
36	72
59	70
8	70
1	60
39	58
53	56

With the following table, we can see that ProductID 40 has the most ordered in Germany, which is **Boston Crab Meat.** Similarly, we could do another join with the “Products” table so we can have the names in our result table above, but the information can also be quickly found.